

Trajectory Generation Model: Building a Simulation Link Between Expert Knowledge and Offline Learning

Arlena Wellßow^{1,2} ^a, Torben Logemann¹ ^b and Eric MSP Veith^{1,2} ^c

¹Adversarial Resilience Learning, Carl von Ossietzky University Oldenburg, Oldenburg, Germany

²OFFIS – Institute for Information Technology, Oldenburg, Germany

fi

Keywords: Modeling, Offline Learning, Simulation, Expert Knowledge.

Abstract: Reinforcement learning has shown its worth in multiple fields (e. g., voltage control, market participation). However, training each agent from scratch leads to relatively long time and high computational power costs. Including expert knowledge in agents is beneficial, as human reasoning is coming up with independent solutions for unusual states of the (less complex) system, and, especially in long-known fields, many strategies are already established and, therefore, learnable for the agent. Using this knowledge allows agents to use these solutions without encountering such situations in the first place. Expert knowledge is usually available only in semi-structured, non-machine-readable forms, such as (mis-) use cases. Also, data containing extreme situations, grid states, or emergencies is usually limited. However, these situations can be described as scenarios in these semi-structured forms. A State machine representing the scenarios' steps can be built from thereon to generate data, which can then be used for offline learning. This paper shows a model and a prototype for such state machines. We implemented this prototype using state machines as internal policies of agents without a learning part for four specified scenarios. Acting according to the actuator setpoints the state machines emit, the agent outputs control the connected simulation. We found that our implemented prototype can generate data in the context of smart grid simulation. These data samples show the specified behavior, cover the search space through variance, and do not include data that violates given constraints.


1 INTRODUCTION


The energy grid, a critical national infrastructure (CNI), evolves into a cyber-physical system called the *smart grid*. This results in a more significantly important adoption of new strategies in grid operation and control Hossain et al. (2023). This is due to the introduction of ICT-based control systems. These components are integrated into the grid to manage the volatile generation and prosumers. They also enable the grid operators to handle grids more efficiently. While providing more usability also leads to a higher risk of errors and potential dangers of cyber attacks. Therefore, a higher risk of failure is given. As this happens, including expert knowledge from grid operators and Information and Communications Technologies (ICT) specialists becomes more critical.


Until now, the risk of a failure within the energy

system was adequately hedged by the redundancy of the physical system (N-1 rule) (De Nooij et al., 2010). With the need for an ever-efficient operation of power grids, which heralds an extensive integration of ICT, this is no longer the case, as ICT integration poses its own risks and redundancy of the whole grid is too cost heavy. The failure of physical systems can lead to the failure of ICT systems. Redundancy is, therefore, still relevant. However, failures of ICT systems can also lead to failures of physical systems. These risks can no longer be solved by redundancies alone and must be mitigated by new technologies (e.g., safe communication protocols, encryption) (Mayer et al., 2020; Schütz et al., 2022).

The power grid transforms, making it highly non-deterministic as an overall system. Due to the high complexity, the introduction of machine learning (ML) systems for optimization, the proliferation of prosumer roles, the emergence of (localized) energy markets, and the high share of distributed renewable energy sources (DERs) to achieve the necessary efficiency goals require testing through extensive sim-

^a  <https://orcid.org/0009-0005-7295-000X>

^b  <https://orcid.org/0000-0002-2673-397X>

^c  <https://orcid.org/0000-0003-2487-7475>

ulations for developing mitigations.

However, this development carries a high effort in terms of time needed for development and financial efforts as this time spent has to be paid (Uslar, 2015; Uslar et al., 2019).

Therefore, previous approaches have employed agent-based systems, often based on Deep Reinforcement Learning (DRL), to generate suitable mitigation strategies for hitherto unknown scenarios (Fischer et al., 2019). However, there is a lot of expert knowledge in energy systems that AI agents do not have by default. For example, agents do not know strategies when starting with random sampling (e.g., using the Soft Actor Critic (SAC) algorithm with random sampling) and do not know about conflicting actors that rewards can not cover. This knowledge needs to be obtained through training to operate as well as or better than human operators. But even if agents are trained for quite some time, they might lack the knowledge for rare occurring situations (for example, misbehavior (Veith et al., 2024)). Sampling efficiency is critical here. If insufficient data is given, even offline learning algorithms cannot learn. This is especially relevant in critical situations that do not occur often enough to be learned by an agent using historical data for training. These situations will be covered by adding scenario data or injection of expert knowledge.

A current best practice in the domain of the electric energy system is the description of system (mis-)behavior utilizing specially designed templates based on the (mis-)use case methodology. This data is usually available only in semi-structured, non-machine-readable forms.

The big picture of our research aims to integrate expert knowledge into AI based experiments to make training of rare situations easier as well as reduce computation expenses.

We have already shown that experiment generation from Misuse Cases is possible and feasible for automation (Veith, 2023a; Wellssow et al., 2024).

In these templates, it is also possible to describe scenarios stepwise. These steps can be converted into a state machine. With this, we can rebuild the whole scenario as a state machine. For now this is done by hand and will be automated in the ongoing research process.

Afterward, this approach sets in. We use the defined state machines, step through them while running a smart grid simulation, and collect their data.

In the big picture shown in Figure 1 can be seen that this data is then used for offline training. As with state machine generation automation and everything before and after these steps, this is not part of this

work and will be shown in later publications.

Our paper is structured as follows: First, we introduce other research work related to this topic. Afterward, needed background information is given, and the framework we used is presented. In the fourth section, we present our model with the fundamental concepts. The subsequent section contains our first prototype that was implemented using this model. In the sixth section, we conclude our findings and discuss our results. Lastly, we give a view of future work.

2 RELATED WORK

The section of related work covers parts of the big picture concerning the research presented in this paper and a subsection regarding offline learning data sampling in the literature.

2.1 Usecase Methodology

The IEC 62559 standard outlines the methodology for use cases, providing a systematic procedure for eliciting and describing them. Following this standard, IEC 62559-2 also furnishes a standardized and structured template for use case descriptions. This template encompasses essential information, including the use case's name, identifier, scope, objectives, conditions, and narrative in natural language. Additional details, such as its relationship with other use cases, prioritization, and related Key Performance Indicators (KPIs), are also included.

The second section of the use case template presents use case diagrams. A technical information summary is provided, listing all acting components and offering a step-by-step analysis for each situation associated with the use case. Processes related to the use case, including lists of exchanged information and requirements, are linked to these analyses. The documentation concludes with the inclusion of common terminologies and specific details.

IEC 62559 is a series of standards widely employed in various fields, as exemplified by Gottschalk et al. (2017).

2.2 Offline Reinforcement Learning for Deep RL

In numerous real-world scenarios, such as large power grids, conducting experiments in the actual environment can be unsafe or computationally expensive, making collecting new data in real-time challenging or risky. This is where *offline reinforcement*

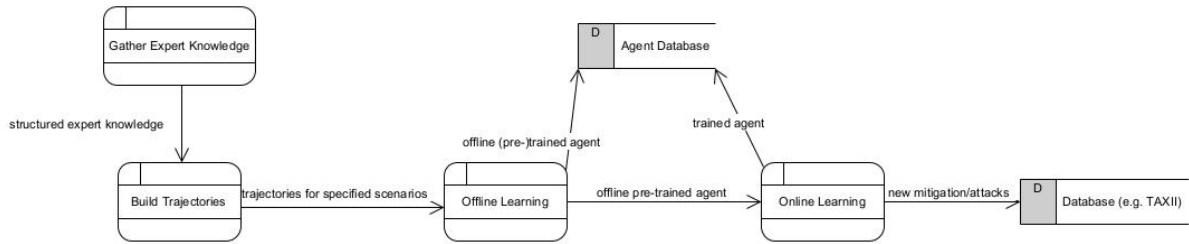


Figure 1: The proposed toolchain of the big picture this work is a part of. [Wellsow et al. (2024)].

learning, also called batch reinforcement learning, becomes relevant (Ernst et al., 2005; Lange et al., 2012; Prudencio et al., 2022; Panaganti et al., 2022).

Offline reinforcement learning enables agents to learn from existing data without direct interaction with the environment. By leveraging pre-existing data that contains observations, actions, and next step observations, agents can enhance their decision-making capabilities efficiently and safely. In this context, safety regards the possible damages caused by faulty operation in the environment. Offline learning is crucial in critical infrastructure cases, such as optimal voltage control in power grids, where making informed decisions without additional data collection is paramount.

Several building blocks and techniques are employed to achieve effective offline reinforcement learning. These are presented in the following paragraphs.

policy constraints (Fujimoto et al., 2018; Kumar et al., 2019; Wu et al., 2019) modify the objective of the learned policy, ensuring adherence to the behavior policy derived from the existing dataset. These constraints enforce a divergence metric between the known and behavioral policies, such as minimizing the discrepancy between their distributions to prevent the agent from deviating too far from the data-driven behavior. Another approach involves estimating the advantage function, which describes the relative improvement of actions, rather than directly estimating the behavioral policy itself (Peng et al., 2019).

Importance sampling (Precup et al., 2000; Jiang and Li, 2016) is another technique employed in offline reinforcement learning, where weights are assigned to samples in the dataset. However, to ensure efficiency, minimizing the variance of these weights is crucial, as their product grows exponentially with the length of trajectories.

Regularization (Haarnoja et al., 2018) in offline RL involves adding a regularization term to the objective function and aims to make the agent’s value function estimates more conservative, preventing over-estimation and discouraging the selection of out-of-distribution (OOD) actions that could be harmful or

suboptimal. An extension of this is *uncertainty estimation*, which allows the agent to dynamically adjust the penalty term based on estimated uncertainty, enabling a flexible trade-off between conservative and naive off-policy deep reinforcement learning methods (Levine et al., 2020).

Model-based methods (Kidambi et al., 2020; Yu et al., 2020) tackle offline reinforcement learning by estimating the dataset’s transition dynamics and reward functions. For example, a surrogate model for a power grid (e.g. as shown by Balduin et al. (2019)) can be learned from a collection of power flow calculations obtained during simulations. However, incorporating uncertainty estimation techniques becomes essential without precise world models, as it helps prevent the agent from transitioning to OOD states.

One-step methods (Brandfonbrener et al., 2021; Kostrikov et al., 2021) in offline reinforcement learning involves collecting multiple states to estimate the action-value function, followed by a single policy improvement step. This differs from iterative actor-critic methods, ensuring that the action-value estimates always represent the distribution of the given dataset, eliminating the need to evaluate actions outside of the data distribution.

Imitation learning (Chen et al., 2020), a class of offline reinforcement learning algorithms, mimics the behavior policy derived from the dataset. It involves techniques like behavior cloning, where the learned policy is trained to replicate the behavior policy using supervised learning. However, this approach is subject to distributional shift, as even small mistakes in the learned policy can lead to compounding errors. To overcome this, imitation learning methods focus on filtering out undesirable behaviors from the dataset and only mimicking the good ones, leveraging value functions and heuristics to select high-quality trajectories.

Another essential technique in offline reinforcement learning is *trajectory optimization* (Janner et al., 2021), which trains a joint state-action model over entire trajectories based on the behavior policy. An optimal action sequence with a good model can be planned from the initial states. The use of sequence

modeling objectives reduces sensitivity to distributional shift, and the model can be optimized using a variety of objectives, such as maximum likelihood or a combination of maximum likelihood and value functions.

2.3 Offline Learning Data Sampling

To effectively train offline learning agents, it is imperative to have a large and diverse enough dataset. The dataset has to represent the learned behavior of the specified scenarios. This is mainly done by sampling data. It can be accomplished by sorting and collecting historical data or by iterative data generation with simulations (Ghysels et al., 2004).

In the smart grid context, data sampling is done in various attempts (Tu et al., 2017; Kayastha et al., 2014; Rhodes et al., 2014).

Two main approaches are random sampling and using Generative adversarial networks (GANs) to generate data. We chose to structure the here shown approach as random sampling has a low sample efficiency rate as well as the convolution problem Gerster et al. (2021), while the usage of GANs is computationally expensive and can't generate data that is better than the data the GAN was trained on Matchev et al. (2022).

3 BACKGROUND

3.1 The PalaestrAI Framework

PalaestrAI¹ is a software stack that enables experimentation, co-simulation, storage and analysis for complex simulations of learning agents in cyber physical systems (CPSs) Veith et al. (2023). The framework provides packages to implement or interface RL agents, environments, and simulators. Its primary focus is on ensuring neat and reproducible execution of experiment runs while coordinating the various parts of these runs and storing their results for later analysis.

One of palaestrAI's major design paradigms is a loose coupling between simulation execution, environment, and agents through a messaging bus. This allows for the implementation of well-known algorithm/rollout-worker splits as they are common in other implementations, but also enables relatively easy extensions of existing algorithms with concepts such as the one presented here. A synchronization between these components happens on the basis of a Ze-

¹<https://gitlab.com/ar12>

roMQ Major Domo Protocol (MDP) implementation, which implements message passing on the basis of a request-reply pattern. In addition, palaestrAI uses a conductor/watchdog concept that separates module management from algorithm logic. This results in a simplified API for developers that facilitates the implementation of new DRL algorithms. This design decision minimizes the learning curve for developers and allows them to focus on algorithmic innovation.

Another benefit of palaestrAI's approach is that co-simulation softwares can be transparently coupled, needing only a minimal API adapter. Our setup uses *mosaik* (Steinbrink et al., 2019) and the MIDAS scenario framework (Balduin et al., 2023) to create the power grid and to synchronize the asset models (e. g., the battery).

Additionally, palaestrAI strongly emphasizes the reproducibility of experiments. The *arsenAI* module, specifically designed for experiment design, ensures that experiments are meticulously laid out in YAML files. These files include critical details such as random seeds, agent specifications, sensor, actuator configurations, and termination conditions for each phase. This approach guarantees that experiments can be replicated while minimizing setup requirements.

Furthermore, palaestrAI allows an unprecedented level of flexibility in experiment design, implementing an actual Design of Experiments (DoE) process. Through the *arsenAI* module, scientists can define a multitude of parameters and factors. This allows for extensive variations in agent combinations, sensor/actuator assignments, and hyperparameters. This versatility enables a comprehensive exploration of diverse scenarios.

3.2 Finite Non-Deterministic State Machines

As defined by Hopcroft et al. (2001) finite non-deterministic state machines are 5-tuples

$$M = (Q, \Sigma, \delta, s_0, F) \quad (1)$$

Where Q is a set of states, Σ is the alphabet the machine is working over, s_0 is the initial state, F is the set of final states, and δ is defined as a relation:

$$\delta : Q \times \Sigma^* \rightarrow 2^Q \quad (2)$$

In general, state machines are used in a wide area of fields like digital event reconstruction (Gladyshev and Patel, 2004), communication modeling (Brand and Zafropulo, 1983), or game development (Jagdaldale, 2021).

In the smart grid context, state machines are used for modeling (as seen in (Kaygusuz et al., 2018)) or

for control in an autonomous, non-learning setting (as seen in (Trigkas et al., 2018)).

For this paper, we use an empty alphabet and do not make use of the final states as we do not want to accept sequences but instead use the structure for data generation in a continuous way.

4 MODEL

We propose a generic state machine model adaptable for different scenarios. While we consider scenarios in the smart grid context, we propose that this concept also applies to other areas.

The state machine is finite. This allows modeling the complete state machine, as we do not consider infinite state sets.

The general concept is depicted in Figure 2 with states named s_i . We use the state machine definition as defined by Hopcroft et al. (2001) with additions to the state and transition definitions.

$$M_{Tg} = (Q, \Sigma, \delta, q_0, F)$$

with

$$(q, (\{c_q\} \in ActuatorSetpoints, \{i_q\} \in TimeStepIntervals)) \in Q$$

$$(i \in Q, n \in Q, \{sc\} \in StepConstraints) \in \delta$$
(3)

Our concept resolves around four significant traits that can be featured in the state machines used in this approach:

Non-Determinism. *Non-Determinism* plays a massive role in this concept, allowing the same state machine to generate differing data. This is because developing differing timelines with one state machine is possible, as the nondeterministic feature allows data generation with state changes at differing time points. Although we consider this to be the most helpful part of generating differing datasets and using it for each presented example, the concept is also usable with deterministic state machines.

State Actuator Constraints C_q . *State Actuator Constraints* operate as the control output. In these constraints, an interval for every output of every asset the agent controls determines the possible setpoints in this state. This interval can be given as an array for one controlled actuator or a combination of a dictionary containing the actuator name as a key and an interval of possible setpoints as a value.

The action spaces of the correlating actuator definitions determine valid setpoints.

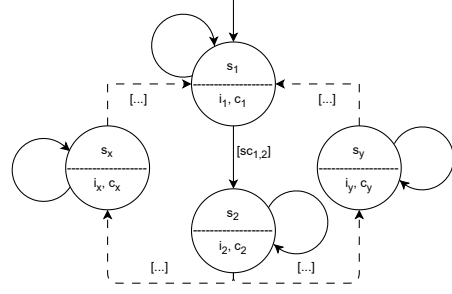


Figure 2: State centered depiction of the proposed state machine concept [own representation].

Time-Step Intervals I_q . *Time-Step Intervals* are a second kind of interval given. This interval determines the timesteps, an agent is set to be in this state with the lower bound being the least amount of steps the agent has to stay in this state and the upper bound being the maximal amount of steps.

Constrained Steps $Sc_{i,N}$. *Constrained Steps* allow to limit a step to certain conditions. This is especially helpful in each situation where a step is not after a certain time in the scenario but if (or if not) a certain condition is satisfied. Constrained steps are defined between two states and are therefore named with the numbers of both states.

4.1 Combination with Online Learning Approaches

To demonstrate how our concept could be connected to online learning agents, we show an exemplary combination with the well-known AWAC offline learning algorithm (Nair et al., 2020) shown in algorithm 1.

Algorithm 1: AWAC-Algorithm as derived from Nair et al. (2020).

```

Dataset  $D = \{(s, a, s', r)_j\}$ ;
Initialize buffer  $\beta = D$ ;
Initialize  $\pi_\theta, Q_\phi$ ;
for iteration  $i = 1, 2, \dots$  do
  Sample batch  $(s, a, s', r) \sim \beta$ ;
   $y = r(s, a) + \gamma \mathbb{E}_{s', a'} [Q_{\phi_{k-1}}(s', a')]$ ;
   $\phi \leftarrow \arg \min_{\phi} \mathbb{E}_D [Q_\phi(s, a) - y]^2$ ;
   $\theta \leftarrow \arg \max_{\theta} \mathbb{E}_{s, a \sim \beta} [\log \pi_\theta(a|s) \exp(\frac{1}{\lambda} A^{\pi_k}(s, a))]$ ;
  if  $i > \text{num\_offline\_steps}$  then
     $\tau_1, \dots, \tau_K \sim p_{\pi_\theta}(\tau)$ ;
     $\beta \leftarrow \beta \cup \{\tau_1, \dots, \tau_K\}$ 
  end
end

```

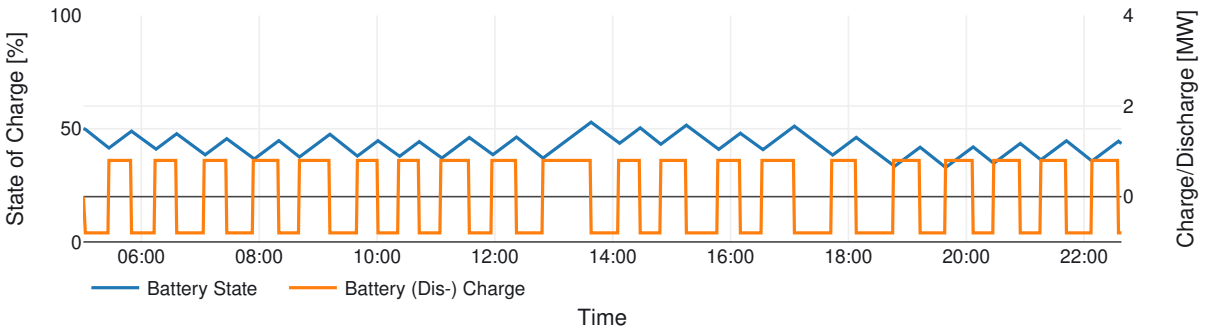


Figure 3: Simple Battery Experiment: Battery Loading State in percent and the agent’s setpoints for (dis-)charging over time. [own representation].

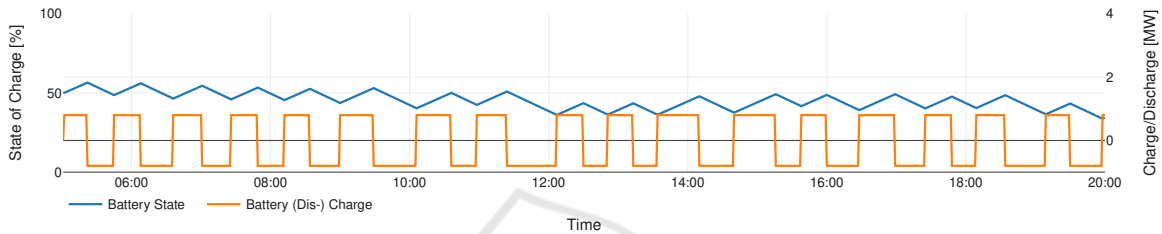


Figure 4: Multiple Actuators Experiment: Battery Loading State in percent and the agents setpoints for (dis-)charging over time. [own representation].

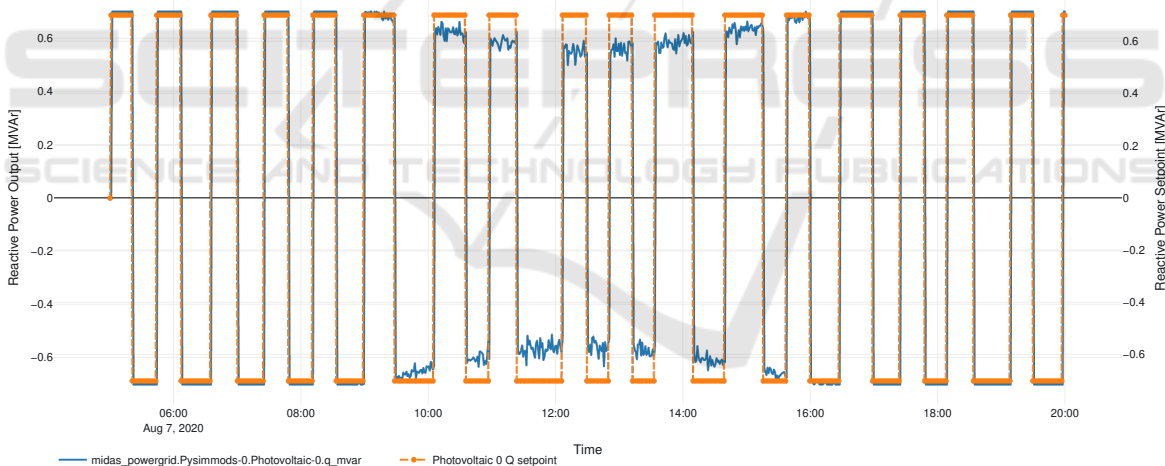


Figure 5: Multiple Actuators Experiment: Setpoints for PV control. [own representation].

As seen in algorithm 2, when combining the two algorithms, our approach runs first as it generates the data needed for the AWAC algorithm. This means, data is generated first and then used for training with the AWAC implementation.

We iterate through the state machine until the maximum of steps is completed. While doing so, we save the states of the simulation before (s) and after (s’) each action (a). $M_{I_g}.advance()$ covers the process of constraint checking and stepping in the state machine. When a constraint for a state transition is not

satisfied, this transition is not callable, and therefore, the state connected with this transition is not reachable. If there are multiple possible transitions, the transition is chosen in a non-deterministic way based on predefined probabilities. In each step, a reward (r) is calculated for the grid state given by the simulation and then stored with the states and action set.

The AWAC implementation can work with the stored data in the same way as with every other dataset in the original implementation.

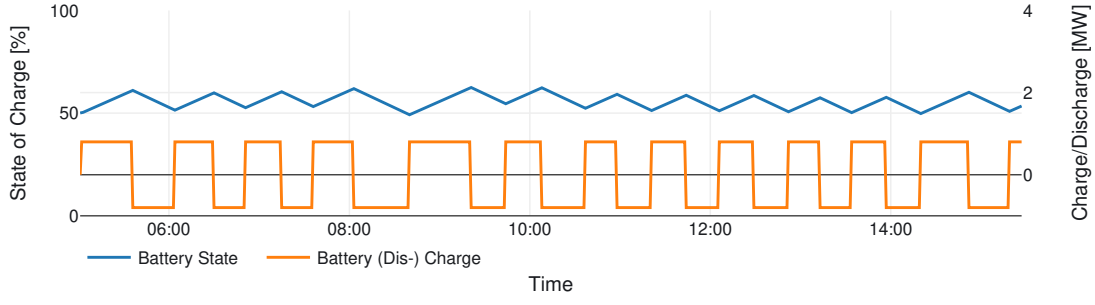


Figure 6: Constrained Step Experiment: Battery Loading State in percent and the agents setpoints for (dis-)charging over time. [own representation].

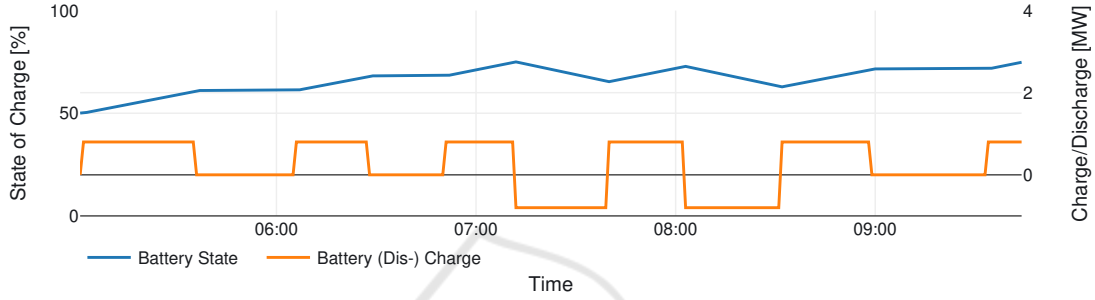


Figure 7: Multiple Paths Experiment: Battery Loading State in percent and the agents setpoints for (dis-)charging over time. [own representation].

Algorithm 2: AWAC-Algorithm combined with our data generation approach.

```

Initialize Simulation  $S$ ;
Statemachine  $M_{I_g} = (Q, \Sigma, \delta, s_0, F)$ ;
maximal_steps =  $x$ ;
for  $j \leq x$  do
     $s = S.state$ ;
     $a \leftarrow \{c \in (M_{I_g}.state, c_{M_{I_g}.state}) \in Q_{M_{I_g}}\}$ ;
     $r \leftarrow \text{reward}(a)$ ;
     $s' = S.step(a)$ ;
    store_to_database( $s, a, s', r$ );
     $M_{I_g}.advance()$ ;
end
Dataset  $D = \{(s, a, s', r)_j\}$  from database;
Initialize buffer  $\beta = D$ ;
Initialize  $\pi_\theta, Q_\phi$ ;
for iteration  $i = 1, 2, \dots$  do
    Sample batch  $(s, a, s', r) \sim \beta$ ;
     $y = r(s, a) + \gamma \mathbb{E}_{s', a'} [Q_{\phi_{k-1}}(s', a')]$ ;
     $\phi \leftarrow \arg \min_{\phi} \mathbb{E}_D [Q_\phi(s, a) - y]^2$ ;
     $\theta \leftarrow \arg \max_{\theta} \mathbb{E}_{s, a \sim \beta} [\log \pi_\theta(a|s) \exp(\frac{1}{\lambda} A^{\tau_k}(s, a))]$ ;
    if  $i > \text{num\_offline\_steps}$  then
         $\tau_1, \dots, \tau_K \sim p_{\pi_\theta}(\tau)$ ;
         $\beta \leftarrow \beta \cup \{\tau_1, \dots, \tau_K\}$ 
    end
end

```

5 IMPLEMENTED PROTOTYPE

The implemented prototype² is constructed as a muscle in palaestrAI. This means it can control given actuators based on sensor information. Normally, each learning agent in palaestrAI contains a muscle and a brain part in which the agent learns from previous steps. We implement the model as a muscle only, as we do want to use the proposing of actions by the muscle. The model works without a brain part, as there is no learning, and the state machine provides the strategy by which the agent works. The use of palaestrAI ensures that the format of the trajectories from the data generation matches the one used for training the agents. Therefore, the data can be used for offline learning afterward without reformatting.

As seen in Figure 8 our prototype contains two classes: The AgentStateMachine, which contains the actual state machine, and the StateMachineAgent, which is wrapped around this state machine. The state machine is built with the python package pytransitions³, which enables all essential state machine functions like stepping, keeping the state, and returning the state the machine is in.

As seen in Figure 9, the agent loops while gener-

²Our code can be found here: <https://gitlab.com/arl2/state-machine-data-generation>

³<https://github.com/pytransitions/transitions>

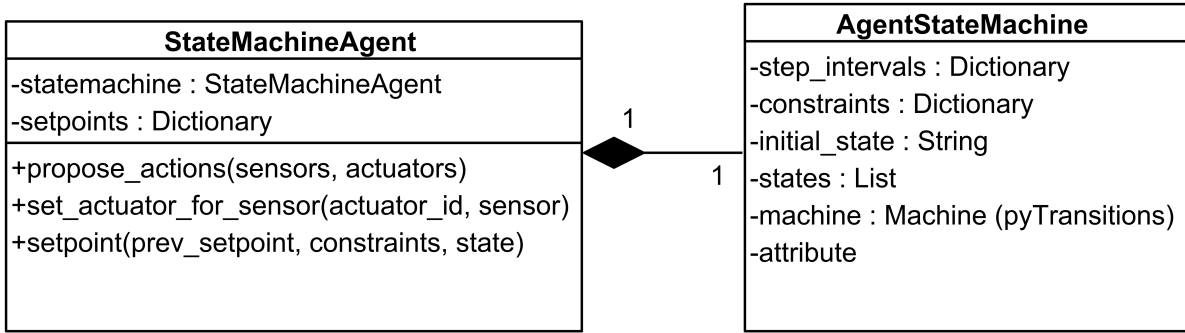


Figure 8: Uml Class diagram of our StateMachineAgent implementation [own representation].

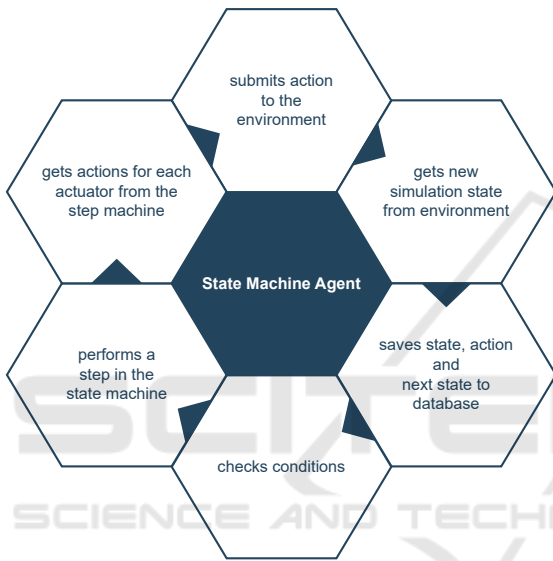


Figure 9: Implemented loop of the StateMachineAgent in an abstract visualization [own representation].

ating data until the maximum amount of steps given before is reached. It starts with getting actions from the state machine for the initial simulation state. Afterward, the actions are submitted to the co-simulated environment, and a new state is reported after the co-simulation has performed those actions. Afterward, the tuple of state, action, next state, and reward is saved to the database and will later serve as part of the sampled data. The StateMachineAgent then checks the conditions (possibly) given for the transitions of the state machine before performing a step. Afterward, actions are provided by the state machine, and the loop begins again.

This is used to iterate over the state machine while the simulation is running. We ran four different experiments with this prototype:

- Simple battery experiment
- Multiple actuators experiment

Table 1: Actuator Mapping for Experiments.

Experiment	Actuator(s)
Simple Battery Experiment	Battery
Multiple Actuators Experiment	Battery, PV
Multiple Paths Experiment	Battery
Constrained Step Experiment	Battery

- Multiple paths experiment
- Constrained step experiment

Simple Battery Experiment. This is our simplest experiment. It contains just two states (S1 and S2) and two kinds of transitions (step and keep) connecting them.

$$M_{rg,A} = (Q, \Sigma, \delta, s1, F) \quad (4)$$

with Q and F as depicted in Figure 10

This state machine is depicted in Figure 10. We used one battery as an actuator to control. The states differ between loading and discharging.

The results of this experiment can be seen in Figure 3. In this chart, the non-deterministic behavior of the state machine is observable through the changes between charging and discharging that happen after differing amounts of timesteps.

Multiple Actuators Experiment. In the second experiment, we gave the agent a second asset to control. States and transitions were kept from the simple battery experiment. For this experiment, we changed our constraint implementation from an array to a dictionary containing the actuator names and their allowed setpoints as an array. Except for this, the state machine was kept the same as in the simple battery experiment. For this experiment, the agent was able to control two assets in the grid: the battery and a photovoltaic asset. Both are independent, co-simulated models to give setpoints to but are modeled within the

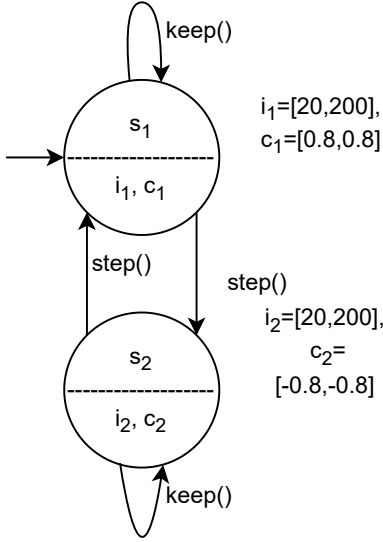


Figure 10: State Machine used in the simple battery experiment and the multiple actuators experiments. [own representation].

same grid and, therefore, impact each other physically in the environment.

As seen in Figure 4 and Figure 5, the values for both assets change according to the changes in the state machine. More states would allow for more detailed control (e.g., changing the setpoints of just one of the assets in a state transition).

Multiple Paths Experiment. For the third experiment, we kept the infrastructure of the simple battery experiment but added another state. This allows a non-deterministic progression of the state machine.

$$M_{Ig,B} = (Q, \Sigma, \delta, s_1, F) \quad (5)$$

with Q and F as depicted in Figure 11

We choose an additional state S3 where the battery is neither charged nor discharged. The adapted state machine is depicted in Figure 11. The additional state is reachable from the initial state S1. The discharging state (S2) can be reached after the next time state S1 is left.

In Figure 7, the non-deterministic state changes after the state of charging are shown in the setpoints. Note, that the battery charge in percent is increasing, as there is non-deterministic change between discharging and keeping the battery charge, while loading is scheduled in every control cycle of the state machine.

Constrained Step Experiment. In the constrained step experiment, the infrastructure and the state machine from the simple battery example are kept. However, we added a constrained step to the state machine,

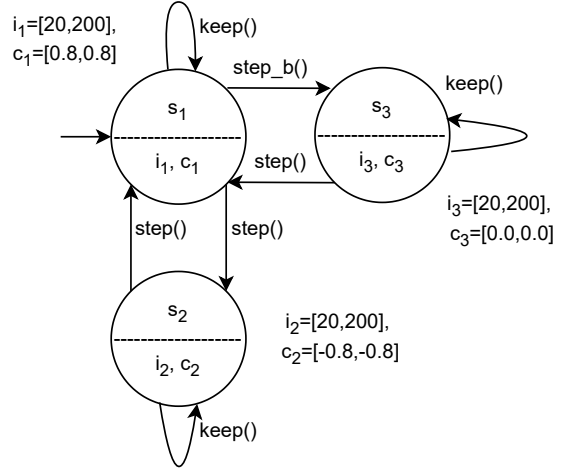


Figure 11: State Machine with additional state used in the multiple paths experiment. [own representation].

where the machine can only step to the discharging state S2 when the battery is charged to at least 50 percent. This is shown in Figure 12.

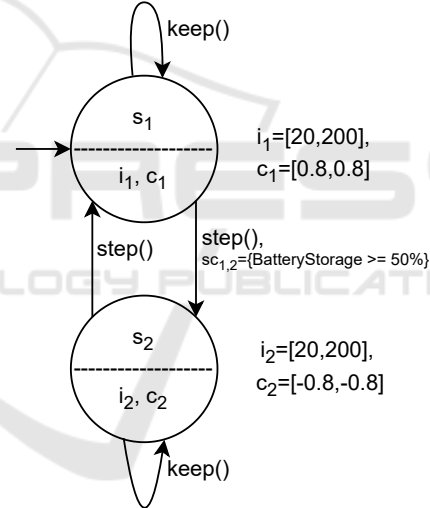


Figure 12: State Machine with constrained step used in the constrained step experiment. [own representation].

Note that in Figure 6 the battery charge in percent is risen over 50 percent in each loading cycle. This is due to the constraint being used in the state machine, as the charging state s1 has to be kept until the constraint is satisfied.

6 CONCLUSION

Within this work, we presented a model for data generation for offline learning using state machines. We also showed an implemented prototype and used it to generate data for four scenarios in the smart grid context. The data generated by the StateMachineAgent

shows the pattern that was represented by the state machines. It also varies due to the non-deterministic nature of the state machines.

Our approach allows the generation of training data containing specific situations and scenarios. It is to be mentioned that this approach also enables the generation of rather significant amounts of (varying) data in a short amount of time.

Our implemented prototype shows that the non-deterministic features and changeable intervals for setpoint determination allow for generating varying data. The Multiple-Actor-Experiment has proven that this concept for data generation also supports multiple actuators. The possibility of varying data through the use of multiple strategies in the state machine that can be executed in a non-deterministic way is shown by the Multiple-Path-Experiment. Further, the Constraint-Step-Experiment indicates that more complex strategies can be encoded in state machines by applying constraints while generating data.

7 FUTURE WORK

Currently, state machines can only be hard coded and not be read dynamically via a specification, e.g., by providing a configuration file. A possible way to do this is by building state machines in a structured way with standardised formats, e.g., with YAML. These could then be given to the StateMachineAgent, which creates an AgentStateMachine containing the information.

Regarding the bigger goal of our research, future work aims to generate State Machines from structured expert knowledge automatically. An expert would enter their valuable knowledge in a semi-structured format, like a misuse case. Afterward, a state machine is generated from these scenario inputs.

Furthermore, an automated state machine generation paired with the presented approach adapted with a YAML read-in, a toolchain could be established, automatically building datasets (or even offline trained agents) whenever new expert knowledge is added.

When the toolchain is enabled, our future work aims to research the limits of its usage. This refers to limiting the fields the toolchain is applicable for and listing the requirements and limitations and the needed amount of available expert knowledge.

It is also planned to include the expert knowledge toolchain in an agent architecture Veith (2023b) that involves neuro-evolutionary algorithms that then continuously revise the rules given in the state machine.

8 FUNDING

The Adversarial Resilience Learning methodology, extended agents, the continued work on palaestrAI, as well as the offline learning development, has been funded by the German Federal Ministry of Education and Research under the project grant *Adversarial Resilience Learning* (01IS22071).

This work has been funded by the German Federal Ministry for Economic Affairs and Climate Action under the project grant *RESili8* (03EI4051A).

REFERENCES

- Balduin, S., Tröschel, M., and Lehnhoff, S. (2019). Towards domain-specific surrogate models for smart grid co-simulation. *Energy Informatics*, 2(1):1–19.
- Balduin, S., Veith, E. M. S. P., and Lehnhoff, S. (2023). Midas: An open-source framework for simulation-based analysis of energy systems. In Wagner, G., Werner, F., and De Rango, F., editors, *Simulation and Modeling Methodologies, Technologies and Applications*, pages 177–194, Cham. Springer International Publishing.
- Brand, D. and Zafiropulo, P. (1983). On communicating finite-state machines. *Journal of the ACM (JACM)*, 30(2):323–342.
- Brandfonbrener, D., Whitney, W., Ranganath, R., and Bruna, J. (2021). Offline rl without off-policy evaluation. *Proc. Adv. Neural Inf. Process. Syst.*, page 4933–4946.
- Chen, X., Wang, Z. Z., Wang, C., Wu, Y., and Ross, K. (2020). Bail: Best-action imitation learning for batch deep reinforcement learning. *Proc. NeurIPS*, page 18353–18363.
- De Nooij, M., Baarsma, B., Bloemhof, G., Slootweg, H., and Dijk, H. (2010). Development and application of a cost–benefit framework for energy reliability: Using probabilistic methods in network planning and regulation to enhance social welfare: The n-1 rule. *Energy Economics*, 32(6):1277–1282.
- Ernst, D., Geurts, P., and Wehenkel, L. (2005). Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6(Apr):503–556.
- Fischer, L., Memmen, J. M., Veith, E. M., and Tröschel, M. (2019). Adversarial resilience learning—towards systemic vulnerability analysis for large and complex systems. In *ENERGY 2019, The Ninth International Conference on Smart Grids, Green Communications and IT Energy-aware Technologies*, number 9, pages 24–32, Athens, Greece. IARIA XPS Press.
- Fujimoto, S., Hoof, H., and Meger, D. (2018). Addressing function approximation error in actor-critic methods. In *Proceedings of the 35th International Conference on Machine Learning*, pages 1587–1596. PMLR. ISSN: 2640-3498.
- Gerster, J., Lehnhoff, S., Sarstedt, M., Hofmann, L., and Veith, E. M. (2021). Comparison of random sampling

- and heuristic optimization-based methods for determining the flexibility potential at vertical system interconnections. In *2021 IEEE PES Innovative Smart Grid Technologies Europe (ISGT Europe)*, pages 1–6.
- Ghysels, E., Santa-Clara, P., and Valkanov, R. (2004). The midas touch: Mixed data sampling regression models.
- Gladyshev, P. and Patel, A. (2004). Finite state machine approach to digital event reconstruction. *Digital Investigation*, 1(2):130–149.
- Gottschalk, M., Uslar, M., and Delfs, C. (2017). *The use case and smart grid architecture model approach: the IEC 62559-2 use case template and the SGAM applied in various domains*. Springer.
- Haarhoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor.
- Hopcroft, J. E., Motwani, R., and Ullman, J. D. (2001). Introduction to automata theory, languages, and computation. *Acm Sigact News*, 32(1):60–65.
- Hossain, R. R., Yin, T., Du, Y., Huang, R., Tan, J., Yu, W., Liu, Y., and Huang, Q. (2023). Efficient learning of power grid voltage control strategies via model-based deep reinforcement learning. *Machine Learning*, pages 1–26.
- Jagdale, D. (2021). Finite state machine in game development. *algorithms*, 10(1).
- Janner, M., Li, Q., and Levine, S. (2021). Offline reinforcement learning as one big sequence modeling problem. *Proc. NeurIPS*, page 1273–1286.
- Jiang, N. and Li, L. (2016). Doubly robust off-policy value evaluation for reinforcement learning. *Proc. ICML*, page 652–661.
- Kayastha, N., Niyato, D., Hossain, E., and Han, Z. (2014). Smart grid sensor data collection, communication, and networking: a tutorial. *Wireless communications and mobile computing*, 14(11):1055–1087.
- Kaygusuz, C., Babun, L., Aksu, H., and Uluagac, A. S. (2018). Detection of compromised smart grid devices with machine learning and convolution techniques. In *2018 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE.
- Kidambi, R., Rajeswaran, A., Netrapalli, P., and Joachims, T. (2020). Morel : Model-based offline reinforcement learning. *Proc. NeurIPS*, page 21810–21823.
- Kostrikov, I., Nair, A., and Levine, S. (2021). Offline reinforcement learning with implicit q-learning.
- Kumar, A., Fu, J., Tucker, G., and Levine, S. (2019). Learning new attack vectors from misuse cases with deep reinforcement learning. *Proc. NeurIPS*, pages 1–11.
- Lange, S., Gabel, T., and Riedmiller, M. (2012). *Reinforcement learning*, chapter Batch reinforcement learning, page pages 45–73. Springer.
- Levine, S., Kumar, A., Tucker, G., and Fu, J. (2020). Offline reinforcement learning: Tutorial, review, and perspectives on open problems.
- Matchev, K. T., Roman, A., and Shyamsundar, P. (2022). Uncertainties associated with gan-generated datasets in high energy physics. *SciPost Physics*, 12(3):104.
- Mayer, C., Brunekreeft, G., Blank-Babazadeh, M., Stark, S., Buchmann, M., Dalheimer, M., et al. (2020). Resilienz digitalisierter energiesysteme. *Blackout-Risiken Verstehen, Stromversorgung Sicher Gestalten*.
- Nair, A., Gupta, A., Dalal, M., and Levine, S. (2020). Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*.
- Panaganti, K., Xu, Z., Kalathil, D., and Ghavamzadeh, M. (2022). Robust reinforcement learning using offline data. *Advances in neural information processing systems*, 35:32211–32224.
- Peng, X. B., Kumar, A., Zhang, G., and Levine, S. (2019). Advantage-weighted regression: Simple and scalable off-policy reinforcement learning.
- Precup, D., Sutton, R., and Singh, S. (2000). Eligibility traces for off-policy policy evaluation. *Proc. ICML*, page 759–766.
- Prudencio, R. F., Maximo, M. R. O. A., and Colombini, E. L. (2022). A survey on offline reinforcement learning: Taxonomy, review, and open problems.
- Rhodes, J. D., Upshaw, C. R., Harris, C. B., Meehan, C. M., Walling, D. A., Navrátil, P. A., Beck, A. L., Nagasawa, K., Fares, R. L., Cole, W. J., et al. (2014). Experimental and data collection methods for a large-scale smart grid deployment: Methods and first results. *Energy*, 65:462–471.
- Schütz, J., Uslar, M., and Clausen, M. (2022). *Digitalisierung. Synthesericht 3 des SINTEG Förderprogramms, Studie im Auftrag des BMWK, Berlin*. Berlin.
- Steinbrink, C., Blank-Babazadeh, M., El-Ama, A., Holly, S., Lüers, B., Nebel-Wenner, M., Ramírez Acosta, R. P., Raub, T., Schwarz, J. S., Stark, S., Nieße, A., and Lehnhoff, S. (2019). Cpes testing with mosaik: Co-simulation planning, execution and analysis. *Applied Sciences*, 9(5).
- Trigkas, D., Ziogou, C., Voutetakis, S., and Papadopoulou, S. (2018). Supervisory control of energy distribution at autonomous res-powered smart-grids using a finite state machine approach. In *2018 5th International Conference on Control, Decision and Information Technologies (CoDIT)*, pages 415–420. IEEE.
- Tu, C., He, X., Shuai, Z., and Jiang, F. (2017). Big data issues in smart grid—a review. *Renewable and Sustainable Energy Reviews*, 79:1099–1107.
- Uslar, M. (2015). Energy Informatics: Definition, State-of-the-art and new horizons. In Kupzog, F., editor, *Proceedings der ComForEn 2015 Vienna*, Wien. TU Wien, OVE Verlag.
- Uslar, M., Rohjans, S., Neureiter, C., Prössl Andrén, F., Velasquez, J., Steinbrink, C., Efthymiou, V., Migliavacca, G., Horsmanheimo, S., Brunner, H., et al. (2019). Applying the smart grid architecture model for designing and validating system-of-systems in the power and energy domain: A european perspective. *Energies*, 12(2):258.
- Veith, E., Balduin, S., Wenninghoff, N., Wolgast, T., Baumann, M., Winkler, D., Hammer, L., Salman, A., Schulz, M., Raeiszadeh, A., Logemann, T., and Wellßow, A. (2023). palaestraAI: A training ground

- for autonomous agents. In *Proceedings of the 37th annual European Simulation and Modelling Conference*. EUROSIS.
- Veith, E., Logemann, T., Berezin, A., WellBow, A., and Balduin, S. (2024). Imitation game: A model-based and imitation learning deep reinforcement learning hybrid. *arXiv preprint arXiv:2404.01794*.
- Veith, E. M. (2023a). An architecture for reliable learning agents in power grids. In *ENERGY 2023, The Thirteenth International Conference on Smart Grids, Green Communications and IT Energy-aware Technologies*, pages 13–16, Barcelona, Spain. IARIA XPS Press.
- Veith, E. M. (2023b). An architecture for reliable learning agents in power grids.
- Wellssow, A., Kohlisch-Posega, J., Veith, E. M. S. P., and Uslar, M. (2024). Threat modeling for ai analysis: Towards the usage of misuse case templates and uml diagrams for ai experiment description and trajectory generation. *IEEA '24*, accepted to be published, New York, NY, USA. Association for Computing Machinery.
- Wu, Y., Tucker, G., and Nachum, O. (2019). Behavior regularized offline reinforcement learning.
- Yu, T., Thomas, G., Yu, L., Ermon, S., Zou, J., Levine, S., Finn, C., and Ma, T. (2020). Mopo: Model-based offline policy optimization. *Proc. NeurIPS*, 33:14129–14142.

